

METHOD AND SYSTEM FOR PROVIDING A MODULAR SERVER ON USB FLASH STORAGE

FIELD OF THE INVENTION

The present invention relates to computer systems, and more particularly to a method and system for providing a server on a generalized computing device.

BACKGROUND OF THE INVENTION

FIG. 1 depicts a generalized computing device ("computing device") 10. The computing device 10 includes at least a CPU 12 and an optional mass storage 18, such as a hard disk. The computing device 10 may also include other features. The computing device depicted in FIG. 1 also includes a memory 14 such as a flash memory, a display 16, an input/output device 20 such as a keyboard, BIOS 22, a network interface 24 and a bus interface 26. Communication to a network (not shown) is carried out through the network interface 24. Similarly, communication to any attached devices (not shown) can be carried out via the bus interface 26. For example, the bus interface 26 could include interfaces for PCI Express, SATA, Ethernet, Infiniband or other serial bus connectors.

The computing device 10 is capable of performing a variety of functions. It is often desirable to utilize the computing device 10 as a server. A server would include additional hardware and/or software that allows the server to serve multiple users. Thus, the server would allow multiple users to share resources, such as printers or the optional mass storage 18 of the computing device 10.

There are a number of conventional methods for allowing the computing device 10 to be used as a server. In general, these conventional methods involve obtaining server software

and installing the software on the computing device 10. The user must then manually set up the desired functions for the server. Alternatively, the computing device 10 could be specially built to function as a server. In either case, ensuring that the computing device 10 can function as a server is expensive. For example, obtaining and installing server software on the computing device 10 or specially building the computing device 10 may cost between \$500 and \$5,000. Moreover, installing the software and tailoring the system to provide the desired individual functions requires a substantial investment of time on the part of the user.

Purpose of Invention

As Universal Serial Bus (USB) becomes a standard communication interface on the PC and digital imaging device, USB-based flash storage system starts proliferating the consumer market. A traditional USB-based flash storage system tends to include an USB Local Control Program, one or more flash memory chips in addition to the USB connector. USB flash storage becomes one of the most popular choices for external removable storage due to its simplicity, high performance and reliability.

If the PC or computing device has the capability in its BIOS to boot from a USB flash storage, it opens up a possibility to incorporate server functionality into a USB flash storage. The server is thus modular and very portable. The actual storage drives on a server are no longer needed to reside on the same physical space with the server itself. It is able to decouple the server from the storage drives completely. The server or storage drives can each evolve or upgrade independent to each other. Being able to easily hot swap the USB flash storage from the PC or computing device, it brings great benefits in service and support to the server itself.

Accordingly, what is needed is a system and method for cheaply and easily allowing the computing device to be used as a server. The present invention addresses such a need.

SUMMARY OF THE INVENTION

5 The present invention provides a method and system for providing a server on a computing device. The computing device includes at least a processor and an optional mass storage device. The method and system comprise providing bus interface logic, providing USB Local Control Program, a flash memory and, preferably, a set of control button connectors, light emitting diodes (LED) connectors and a liquid crystal display (LCD) connector. The USB Local Control Program is coupled with the bus interface logic and the 10 memory. The bus interface logic interacts with the computing device and allows the computing device to detect the system. The USB Local Control Program boots up the server and prepares the computing device for use as the server. The memory stores a server image for the server, which is provided to the computing device using the USB Local Control 15 Program. The control button connectors allow the server to be turned on, shut down gracefully, or restored to its initial state, by a single press of buttons connected to these connectors. The LED and LCD connectors allow the system status to be displayed or shown.

 According to the system and method disclosed herein, the present invention provides an inexpensive, easy to use mechanism for allowing the computing device to be used as a 20 server.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a conventional computing device.

FIG. 2 is a high level block diagram of a system in accordance with the present invention for allowing the computing device to be used as a server.

FIG. 3 is a block diagram of one embodiment of the Local Control Program of the system in accordance with the present invention for allowing the computing device to be used as a server.

FIG. 4 is a diagram of one embodiment of the image of the server stored in the memory of the system in accordance with the present invention for allowing the computing device to be used as a server.

FIG. 5 is a more detailed block diagram of one embodiment of the other control logic in the system in accordance with the present invention for allowing the computing device to be used as a server.

FIG. 6 is a flow chart of one embodiment of a method in accordance with the present invention for utilizing the system in accordance with the present invention to allow the computing device to be used as a server.

FIG. 7 is a flow chart of one embodiment of a method for using one-button shut down interrupt logic as a feature of the system in accordance with the present invention for allowing the computing device to be used as a server.

FIG. 8 is a flow chart of one embodiment of a method for a shut down interrupt routine in the system in accordance with the present invention for allowing the computing device to be used as a server.

FIG. 9 is a flow chart of one embodiment of a method for using one-button Init interrupt logic as a feature of the system in accordance with the present invention for allowing the computing device to be used as a server.

FIG. 10 is a flow chart of one embodiment of a method for an Init interrupt routine in the system in accordance with the present invention for allowing the computing device to be used as a server.

FIG. 11 is a flow chart of one embodiment of a method for using one-button power on control logic as a feature of the system in accordance with the present invention for allowing the computing device to be used as a server.

DETAILED DESCRIPTION

The present invention relates to computer systems, and more particularly to a method and system for providing a server on a generalized computing device. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment and the generic principles and features described herein will be readily apparent to those skilled in the art. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

The present invention relates to an improvement in computer systems. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment will be readily apparent to those skilled in the art and the generic principles herein may be applied to other embodiments. Thus, the present invention is not intended to be limited to the embodiment shown, but is to be accorded the widest scope consistent with the principles and features described herein.

The present invention provides a method and system for providing a modular server on a board. The server-on-a-USB is installed on a computing device. The method and system include providing bus interface logic, providing a USB Local Control Program, flash memory and, preferably, a set of control button connectors, light emitting diodes (LED) connectors and a liquid crystal display (LCD) connector. The USB Local Control Program is coupled with the bus interface logic and the flash memory. The bus interface logic interacts with the computing device and allows computing device to detect the server board. The USB Local Control Program boots up the server and prepares the computing device for use as the server. The flash memory stores a server image for the server, which is provided to the computing device using the USB Local Control Program. The control button connectors allow the server to be turned on, shut down gracefully, or restored to its initial state, by a single press of buttons connected to these connectors. The LED and LCD connectors allow the system status to be displayed or shown.

The present invention will be described in terms of a particular computing device and a system having certain components. However, one of ordinary skill in the art will readily recognize that this method and system will operate effectively for other computing devices and other systems having other components performing substantially the same functions.

To more particularly illustrate the method and system in accordance with the present invention, refer now to FIG. 2, depicting a high-level block diagram of a system 100 in accordance with the present invention for allowing the computing device to be used as a server. The system 100 is to be used in conjunction with a computing device such as the computing device 10. The system 100 includes bus interface logic 102, USB Local Control Program 104, memory 106 and, in a preferred embodiment, other control logic 108 and

connectors 109. The components 102, 104, 106, 108 and 109 of the system 100 are preferably integrated into a single board that can be plugged into the computing device 10.

The system 100 is also preferably used in conjunction with a system having a generic user interface, such as Windows 2000.RTM. operating system. The system 100 attaches to the computing device 10 via the bus interface logic 102 and bus interface 103 of the system 100 and the bus interface 26 of the computing device 10. In operation, the computing device 10 detects the system 100 through the bus interface logic 102, using the bus protocols of the computing device 10. The USB Local Control Program 104 boots up the server and prepares the computing device for use as the server.

The memory 106 includes a server image 110 for the server being provided by the system 100. Preferably, the server image 110 is compressed and stored on the memory 106. The server image 110 is preferably loaded onto the computing device 10 and boots up, as discussed below. Once booted up, the server image 10 allows the computing device 10 to function as a server. In addition, the system 100 also includes the other control logic 108. In a preferred embodiment, the other control logic 108 is managed by the USB Local Control Program 104. The connectors 109 preferably include an Init connector 112, a shut-down connector 114, a power control connector 116, a status LED connector 118, a DC power LED connector 120 and a LCD display connector 122. However, in another embodiment, the other control logic 108 could include other components. The connectors 109 can be coupled to LEDs (not shown) and an LCD display (not shown) for the board. The connectors 109 are controlled using the other control logic 108.

FIG. 3 depicts one embodiment of the Local Control Program 104. The Local Control Program 104 includes a system initialization and testing block 130, a local Control

Program run-time main program 132, an LCD display driver 134, a memory driver 136, a shut-down interrupt service routine 138, and an Init service routine 140. The drivers 134 and 136 are used to drive the display 122 and the memory 106. The shut-down interrupt service routine 138 and Init service routine 140 are used in conjunction with the other control logic 108 described below.

Referring to FIGS. 2 and 3, in operation, once the computing device 10 detects the presence of the system 100, the Local Control Program 104 is activated. The Local Control Program 104 preferably connects with the BIOS 22 and begins controlling the computing device 10. The Local Control Program 104 preferably performs tests on the system 100 to ensure that the system 100 can control the functions of the computing device 10 as desired. For example, the USB Local Control Program 104 ensures that the display, memory and other input/output devices can be controlled. For example, in a preferred embodiment, the hardware identification of the flash memory 106 is read to determine the size of the memory 106. The system initialization and testing block 130 preferably performs the testing functions. An Ethernet MAC address of the computing device 10 is also preferably read to ensure that security and personalization of the computing device 10 is preserved. In a preferred embodiment, an identification for the system 100 is read by the USB Local Control Program 104 to determine a version of the system 100. The USB Local Control Program 104 also preferably establishes a unique personalized key, discussed below. The USB Local Control Program 104 establishes a boot-up sequence on the computing device 10. The memory 106 is then mounted and boots up. The server image 110 is then extracted from the memory 106 using the unique personalized key. Without the key, the server image preferably cannot extract and utilize the server image 110.

FIG. 4 is a diagram of one embodiment of the images for the server stored in the memory 106. The server image 110 includes a default field configurable and field upgradeable bitmap image 141 of the other control logic 108, an active field configurable and field upgradeable bitmap image 142 of the other control logic 108, a default compressed server image 143, an active server image 144, a default flash drive boot-up image 145 and an active flash drive boot-up image 146. The bitmaps 141 and 142 indicate the default and actual (active) bitmap images for the control logic to allow the server to track and utilize the control logic 108.

The compressed server images 143 and 144 are the default and actual (active) server images for loading onto the computing device 10. The active server image 144 thus corresponds to the server image 110, depicted in FIG. 2, that is loaded onto the computing device. The flash drive images 145 and 146 are the default and actual (active) boot-up images of the flash memory 106.

Once the server image 110 is loaded on the computing device 10, the computing device 10 can function as a server. Furthermore, the defaults can be restored, for example in an Init interrupt, described below in FIG. 10, using the defaults 141, 143 and 145. The shut-down interrupt service routine 138 and Init service routine 140 can optionally reside in the server image of 110 as well.

FIG. 5 is a more detailed block diagram of one embodiment of the other control logic 108 in the system 100 in accordance with the present invention for allowing the computing device to be used as a server. The other control logic 108 includes a Local Control Program 104 address decode and control 150, a flash memory address decode and control 152, an LCD address decode and control 154, one button shut-down interrupt logic 156, ID, status

and control decode 158 and one button Init interrupt logic 160. These blocks are used to provide the additional functions, described below, such as a one button shut down and Init interrupt.

FIG. 6 is a flow chart of one embodiment of a method 200 in accordance with the present invention for using the system 100. The method 200 preferably commences after the computing device 10 has found the system 100. The method 200 is described in the context of the components depicted in FIGS. 1-5. Referring to FIGS. 1-6, the USB Local Control Program 104 is automatically coupled with the BIOS 22 of the computing device 10, via step 202. The USB Local Control Program 104 takes control of the computing device 10, via step 204. The functions of the system 100 are tested, via step 206.

It is determined whether the test(s) performed in step 206 indicate that the system 100 is functioning properly, via step 208. If not, then the method 200 terminates, via step 220. If it is determined that the system 100 runs properly, then the memory 106 is mounted on the computing device 10, via step 210. The boot up of the computing device 10 is then performed from the memory 106 that was just mounted, via step 212. The server image 110 is found, decompressed if necessary, via step 214. It is determined whether the functions of the method 200 were properly performed, via step 216. If so, then control is passed to the server, via step 218. Otherwise, the method 200 ends at step 220.

Thus, the method 200 and system 100 allow the computing device 10 to be used as a server. Because most of the method 200 is performed automatically, the user need not manually configure the computing device 10. Instead, the user merely plugs in the board on which the system 100 is integrated. Thus, the process used to allow a computing device 10 to be used as a server is simplified. Moreover, the system 100 is relatively inexpensive, often

costing on the order of less than \$25 in quantity. Thus, the computing device 10 can be turned into a server relatively cheaply and easily.

The system 100 also preferably uses the other controls 108 and connectors 109 to provide other functions in the server. FIG. 7 depicts one embodiment of a method 220 for utilizing one button shut-down interrupt logic 156 and the shut-down connector 114. The one button shut-down interrupt logic 156 waits for input, via step 222. In a preferred embodiment, the input includes a push button (not shown) being depressed for a particular time. It is determined whether shut-down input was received, via step 224. If not then step 222 is returned to. Otherwise, clock sampling is performed to allow for hardware de-bounce, via step 226. It is determined whether the input was valid shut-down input, via step 228. In a preferred embodiment, valid shut-down input includes the push button being depressed for a particular time.

If the input was not valid, then step 222 is returned to. Otherwise, further shut-down interrupts are inhibited, via step 230. Step 230 ensures that the method 220 can be completed for the valid shut down input already provided. A shut down interrupt to the server is then generated, via step 232. A method for generating such an interrupt is described below with respect to FIG. 8. The main system power is then shut down and the system 100 is put into stand-by mode, via step 234. Thus, the system 100 can be shut down using a single press of a button. A user can, therefore, shut down the server provided using the system 100 relatively quickly and easily, through the use of a single button.

FIG. 8 is a flow chart of one embodiment of a method 240 for a shut down interrupt routine in the system 100 in accordance with the present invention. The method 240 is preferably implemented in conjunction with the one button shut-down interrupt logic 156. A

shut-down interrupt service routine entry is provided, via step 242. A status port of the system 100 is read, via step 244. The status port of the system 100 indicates whether a shut down is pending. It is determined whether a shut down is pending, via step 246. If not, then the method 240 is terminated, via step 254. Otherwise, a shut down sequence for the server is initiated, via step 248. The server is then shut down, via step 250. The main power to the system 100 is then shut down and the system 100 is put into standby mode, via step 252. Thus, the system 100 can be shut down relatively simply and easily.

FIG. 9 is a flow chart of one embodiment of a method 260 for using one-button Init interrupt logic a feature of the system 100 in accordance with the present invention. The method 260 is used in conjunction with the one button Init interrupt logic 160 and the Init connector 112. The one button Init interrupt logic 160 waits for connector input, via step 262. The connector input is preferably a push button (not shown) being depressed. It is determined whether Init input is received, via step 264. If not, step 262 is returned to. Otherwise, clock sampling is performed to allow for hardware de-bounce, via step 266. It is determined whether the Init input received is valid, via step 268. If not, step 262 is returned to. Otherwise, further Init interrupts are inhibited, via step 270. Step 270 ensures that the method 260 can be completed for valid Init input already received. An Init interrupt to the server is then generated, via step 272. The server is thus restored to its default state using the method 260. The return to the default state is preferably found in the default server image 143 residing on the memory 106.

FIG. 10 is a flow chart of one embodiment of a method 280 for an Init interrupt routine in the system 100 in accordance with the present invention. The method 280 is preferably used for performing the step 272 of the method 260.

A Init interrupt service routine entry is provided, via step 282. A status port of the system 100 is read, via step 284. The status port of the system 100 indicates whether an initialization is pending. It is determined whether an initialization is pending, via step 286. If not, then the method 280 is terminated, via step 290. Otherwise, the server is restored to its default state, via step 288. Thus, the system 100 can be initialized relatively simply and easily, by a push of a button by a user.

FIG. 11 is a flow chart of one embodiment of a method 300 for using one-button shut down and power on control logic as a feature of the system 100. The method 300 is preferably performed using the power on control connector 116 and the shut-down connector 114. The power control connector (not shown) of the computing device 10 is coupled with a power-on connector 116, via step 302. The AC power to the system 100 is then turned on, the DC power to the system 100 turned off, and the server of the system 100 placed in standby mode, via step 304. It is determined whether the shut-down button has been depressed, via step 306. If not, step 306 is returned to.

Otherwise, DC power for the system 100 is turned on and the system 100 boots up, via step 308. It is then determined whether power is to be disabled, via step 310. If so, then the power on is asserted, via step 314 and the system DC power turned off via step 324. If power is not to be disabled, then it is determined whether the shut-down interrupt is to be enabled, via step 312. If not, it is determined whether the shut-down button has been pressed, via step 322. If so, then the system DC power is turned off, via step 324. Otherwise, the method returns to step 310. If it is determined in step 312 that the shut-down interrupt is to be enabled, power on is de-asserted, via step 316. It is then determined whether the shut-down button has been pressed, via step 318. Preferably, step 318 determines whether the

shut-down button has been pressed for a particular amount of time. If not, then the method returns to step 310. Otherwise, the shutdown input is generated, via step 320 and step 310 returned to.

Thus, using the method 300, the shut-down button can be used in different ways. If the shut down button is pressed prior to a shut-down interrupt being enabled, then the method 300 allows the DC power to the system 100 to be turned off. If, however, the shutdown interrupt was enabled, as determined in step 312, prior to the shut-down button being pressed, then the shut down input generated in step 320 and the system 100 can be shut down using the method 220. Thus, using the method 300, the shut-down button can be used either to turn off the DC power to the system or to shut down the system 100. Thus, using the methods 220, 240, 260, 280 and 300, additional functions can be provided using the system 100.

A method and system has been disclosed for allowing a computing device to be used as a server. Software written according to the present invention is to be stored in some form of computer-readable medium, such as memory, CD-ROM or transmitted over a network, and executed by a processor. Consequently, a computer-readable medium is intended to include a computer readable signal which, for example, may be transmitted over a network.

Accordingly, a system and method in accordance with the present invention applies to a variety of mass storage devices such as Serial ATA FLASH hard drive, IDE FLASH hard drive, SCSI FLASH hard drive and Ethernet FLASH hard drive. In addition, a FLASH controller in accordance with the present invention also applies to FLASH memory cards such as Express Card, Mini PCI Express Card, Secure Digital Card, Multi Media Card, Memory Stick Card and Compact FLASH card. Finally, a system in accordance with the

present invention also applies to the other serial buses such as PCI Express bus, Serial ATA bus, IEEE 1394 bus and Ethernet bus. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.

Although the present invention has been described in accordance with the
5 embodiments shown, one of ordinary skill in the art will readily recognize that there could be variations to the embodiments and those variations would be within the spirit and scope of the present invention. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.